CONSTRAINT instruction

The **CONSTRAINT** instruction creates a constraint.

Syntax

```
CONSTRAINT eventClause constraintExpr [CHECKED [BY propertyId1, ..., propertyIdN]] MESSAGE messageExpr;
```

Description

The **CONSTRAINT** instruction creates a constraint. If the constraint is violated, the user will be shown the message defined in the instruction.

Also, by using the **CHECKED** option you can use the constraint when showing dialogs for changing properties whose values may violate the constraint if changed. In this instance an additional filter will be set in the dialog so that, when the property value changes, the constraint is not violated. If it is necessary to limit the set of properties for which the above filtering will be performed, the list of properties can be specified after the keyword **BY** .

Creating a constraint is pretty similar to the following instruction:

```
constraintProperty = constraintExpr;
WHEN eventClause [=GROUP MAX constraintProperty()]() DO {
        PRINT outConstraintPropertyForm MESSAGE NOWAIT;
        CANCEL;
}
```

but it also has a number of advantages.

Parameters

eventClause

Event description block. Describes the event upon occurrence of which the created constraint will be checked.

constraintExpr

An expression whose value is a condition for the constraint being created. If the obtained property does not contain the **PREV** operator, the platform automatically wraps it into the **SET**operator.

propertyId1, ..., propertyIdN

List of property IDs. When showing change dialog for each property in that list, options that violate the created constraint will be filtered.

messageExpr

An expression whose value is shown as a message to the user when the set constraint is violated. It may be either a string literal or a property without parameters.

Examples

```
// balance not less than 0
CONSTRAINT balance(Sku s, Stock st) < 0
    MESSAGE 'The balance cannot be negative for ' + (GROUP CONCAT 'Product: ' + name
(Sku ss) + ' Warehouse: ' + name(Stock sst), '\n' IF SET(balance(ss, sst) < 0));

barcode = DATA STRING[15] (Sku);
// "emulation" security policy
CONSTRAINT DROPCHANGED(barcode(Sku s)) AND name(currentUser()) != 'admin' MESSAGE 'Only
the administrator is allowed to change the barcode for an already created product';

sku = DATA Sku (OrderDetail);
in = DATA BOOLEAN (Sku, Customer);

CONSTRAINT sku(OrderDetail d) AND NOT in(sku(d), customer(order(d)))
    CHECKED BY sku[OrderDetail] // a filter by available sku when selecting an item for
an order line will be applied
    MESSAGE 'In the order, a product unavailable to the user is selected for the
selected customer';
```